SFA Modernization Partner

Task Order 22
CIO Technical Support Team

Deliverable 22.1.1 Project Assessment

Introduction

The purpose of this deliverable is to document the lessons learned relative to the School Portal and IFAP Conversion applications under Task Order 12. The lessons learned are derived from the issues encountered by the project team during various phases of the project. The issues are categorized in five categories:

- Project Management
- Development Life Cycle
- Integration of New Technologies
- Documentation and Processes, and
- Training

Each category identifies notable issues and the lessons learned to plan for, and thereby avoid, these types of variables during software development efforts. This document is not intended to replace the issues tracking log utilized during the project nor is it a complete listing of all issues. Its intent is to avoid similar issues on subsequent development efforts and facilitate successful achievement of project goals. A summary of all the is sues and related lessons learned is attached as an appendix to this document.

Project Management

Issues Identified:

- **Definition of Project Scope**
- **Monitoring of Management Processes**
- **Key Members within Project Organization**

Definition of Project Scope:

The School Portal and IFAP applications, from the very onset of the project (under task order 12) have been treated as a singular project. Multiple system development efforts should be planned separately and include system integration activities for the combined initiative. This type of planning will allow achieving an optimum balance between the project goal, and the level of effort, scope and schedule of each project. Such planning will also identify dependencies that can otherwise go unnoticed, can demand the appropriate management attention and aid in the proper assignment of skilled resources. Furthermore, potential risks can be better analyzed and mitigated and issues can be better managed with individual plans for system development initiatives and the subsequent effort to integrate them. The resulting two-step testing (at individual system level and the combined integrated level) will also contribute to a more robust user application.

Monitoring of Management Processes:

This issue revolves around the capability to develop a detailed project plan and the inherent input sources and control mechanisms associated with a plan. Of paramount importance is a clear understanding of the customer's objective(s); including among others technical functionality, system schedule and business impact. It may be necessary to plan a brief amount of time initially to validate this understanding among all the stakeholders. The detailed project plan that follows should possess milestones that can be monitored during the activities leading to it. The inability to obtain feedback during the process can rapidly lead to wasted time and resources. Conversely, the lack of planning to account for addressing feedback can also impact the plan (e.g., planning adequate time to address results from testing phases). Estimates for completing activities should be based on both quantitative and qualitative sources of data. The activities themselves should include all potential activities necessary to achieve success. Schedule, resource or other constraints should be documented to identify the activities that may need to be shortened or curtailed to meet customer goals. For all such situations, a risk management plan should be agreed to with the customer and appropriate mitigation strategies implemented. The project plan should account for these risks since by definition, risks are potential problems that may occur given certain circumstances. And finally, progress relative to the plans as well as changes in planning assumptions should be accurately and clearly documented, as they occur, for stakeholders to understand. For large projects, a project repository accessible to appropriate members can be beneficial.

Key Members within Project Organization:

For system development initiatives, key members of each project (at individual system level as well as for integration activities) should ideally include a single point of contact such as a project manager, technical or software architect, change control and configuration manager, software development manager, business analyst, test manager and LAN administrator at the very least. Additionally, where automated tools are utilized, resources should be accessible and fluent in both the knowledge and skills required for the activity and possess the requisite experience with the automated tool (e.g., the use of Rational software for testing). Such organization will lead to an effective span of control and strong communication channels while maintaining clear areas of responsibility. An impact assessment should be performed for any change in the project organization.

Lessons Learned:

- **Develop comprehensive project plans identifying entry and exit milestones for each project phase; including adequate time for addressing test results**
- **Develop project plans using a disciplined approach that lends itself to quantitative measurement for progress**
- **Utilize efficient management processes for effective communication and management of expectations**
- **Organize and staff project teams identifying clear roles and responsibilities for all key activities**
- **Utilization of resources skilled in specific responsibilities assigned to them as well as the experience necessary for required automated tools**

Development Life Cycle

Issues Identified:

- **Business Requirements vs. Technical Specifications**
- **Comprehensive Change Management**
- **Disciplined COTS Development Processes**

Business Requirements vs. Technical Specifications:

All project architectures dictate the use of currently available or newly introduced technologies within an enterprise. In either case, user or customer requirements are generally in the form of business requirements. For system development efforts to occur successfully, they should be translated to detailed technical specifications. This critical step ensure two separate objectives; first, it provides clear direction to developers and helps generate comprehensive test cases and, secondly this step helps initiate a dialogue between technical architects and business users to validate the technological feasibility of delivering functionality or the need to curtail functionality and plan for future enhancements later. This is particularly critical in environments where technology is new to an organization or is being planned for the future. Clear comprehension of what is required by business users and what can practically be delivered within the constraints of the project can only strengthen the process to successful completion. This process, when documented clearly, provides a degree of traceability that helps manage the requirements throughout the product life cycle (development, integration, testing, operations and enhancements).

Comprehensive Change Management:

Change management processes should be comprehensive in their scope to address software configuration management, as well as environmental variables. The latter include areas such as hardware platforms and associated operating systems, COTS (commercial off the shelf) software being utilized as part of the architecture, as well as multiple environments for different phases of a project. Any change, minor or not, directly impacts the behavior of systems which therefore requires strict adherence to a disciplined process for change management. Finally, communication of any deviation from known (and base-lined) configurations should be communicated across the team. In particular, when systems are being developed in one (development) environment, component tested in another (testing environment), system tested in another (staging environment) and deployed in yet another (production environment). Where appropriate, impact assessments should be the natural source for impacts on project schedule, current or planned resources and current or planned activities.

Disciplined COTS Development Processes:

The use of COTS products in a system development initiative requires the rigor of understanding the COTS product architecture, APIs (application programming interfaces), recommended standards and limitations. Unless certified by the vendor, developers will need to address integration requirements among multiple COTS products, where utilized. In the event that documentation is unavailable from the vendor(s) the project should account for the schedule and resource requirements to achieve the desired level of integration. In general, the larger the number of COTS products being utilized for a single solution, the larger the integration effort. Utilizing a COTS product outside of its published functionality can lead to serious maintenance and enhancement issues in the future.

Lessons Learned:

- **Implement requirements management processes that enable all future stages of the project**
- **Use change management processes encompassing the entire system, i.e., software, hardware, integration, training, etc.**
- **Institute appropriate software development processes that support effective system development using 4GL (4th generation languages), COTS and COTS (or industry supported) APIs (application programming interfaces)**

Integration of New Technologies

Issues Identified:

- **Concurrent Deployment of New Technologies**
- **Infrastructure and Connectivity**
- **Standards Supporting Technology Configuration**

Concurrent Deployment of New Technologies:

While certain dependencies can be minimized for optimal project progress, it is important to identify the dependencies that are not removable. Deploying new technologies while developing software will require an additional integration test at the conclusion of the effort. Furthermore, it is likely that some level of software rework will be required to properly align the software towards the installed infrastructure. Complications can arise in cases where the technologies are being deployed concurrently to multiple environments such as those for development, testing, staging and production.

Infrastructure and Connectivity:

As noted earlier in this document, projects should adequately plan lead times and effort for the necessary infrastructure to support the development effort. Any lack of, or delay in obtaining, connectivity to the environment will further impact the achievement of overall project goals. Risks may include hardware procurement delays, telecommunications problems, unavailability of resources, untested solutions, database placement conflicts, etc. Decisions related to initiating development efforts in remote locations and subsequently migrating to the "official" location will also require planning with a likely impact on schedule as well as some level of software rework.

Standards Supporting Technology Configuration:

Once configured and installed, the architecture of the development and production environments should be communicated to project members. Additionally, if available, any standards unique to the organizational environment should also be communicated and impact assessments performed to determine risks or issues. The presence of a technical or software architect as a key member of the project organization becomes increasingly important in such instances. The technical or software architect can provide valuable analyses of the variances among the current, planned and future environments.

Lessons Learned:

- **The implementation platform for a system should be base-lined and agreed to early in the development life cycle**
- **Software development should always occur in the environment within which the systems will operate**
- **A dedicated technical architect should be a key member of the software development team**

Documentation and Processes

Issues Identified:

- **Unified Application Development Processes**
- **Documented Operational Processes**

Unified Application Development Processes:

It is not uncommon to have multiple teams supporting a single project. In instances where the multiple teams are from different organizations, there is a risk that the application development methodologies and processes utilized will be different. It is important to effectively manage this risk as part of the project. The mitigation strategy should focus on outcomes and not necessarily the dictate of using a particular methodology. Where processes are available for use across the teams, they should be clearly documented and communicated. The important aspect is the ability for project management personnel to effectively monitor progress and identify risks and issues that can be mitigated prior to becoming problems. Failure to plan for these activities will likely lead to unnecessary misinterpretation and confusing outcomes. In areas where processes are not available, the project should develop a set for their use. The processes should encompass web standards, security standards, software life cycle standards and communication standards at a minimum.

Documented Operational Processes:

All operational processes, whether currently available or being developed by the project team, should be clearly documented, communicated and maintained. These processes can include those for operational readiness, migration processes, etc. The inability to maintain the operational processes is a serious risk which can lead to significant software rework to meet project goals. As such, these should be planned for within the project schedule.

Lessons Learned:

- **Clearly communicated outcomes based on common processes should be utilized by all teams comprising a system**
- **Operational processes should be documented and maintained current for development teams to plan ahead**

Training

Issues Identified:

- **Training of Multiple Categories of Users**
- **Introduction of Operational Team(s)**

<u>Training of Multiple Categories of Users</u>:
Training for all potential users should be planned for, developed, and provided in sufficient time and detail to ensure better adoption of the new systems and the associated business processes. In general, the larger the amount of functional or business change, the higher the amount and frequency of training that will be required. Additionally, the project may need to assess the types or categories of users that will require training. The multiple categories of user training may be for end users, administrators, maintenance personnel and developers (where required). Each of these categories of users require different functionality and have differing expectations. As such, a comprehensive training plan should be developed to promote the efficient adoption of the systems and the new business processes by which job functions will be performed.

<u>Introduction of Operational Team(s):</u>
The introduction of operation teams should be initiated prior to migration of systems to the production environment. This will facilitate a comprehensive knowledge transfer of business functions, business processes, system maintenance requirements and future enhancements or requirements.

Lessons Learned:

- **All potential users of a system should be identified early during the project and appropriate training plans developed accordingly**
- **Operational teams tasked with supporting a new application should be an integral part of the software migration process from development to production**

APPENDIX:

<u>SUMMARY OF ISSUES & LESSONS LEARNED</u>

	I S S U E C A T E G O R I E S				
	PROJECT MANAGEMENT	**DEVELOPMENT LIFE CYCLE**	**INTEGRATION OF NEW TECHNOLOGY**	**DOCUMENTATION OF PROCESSES**	**TRAINING**
ISSUES	*Definition of Project Scope*	*Business Requirements vs. Technical Specifications*	*Concurrent Deployment of New Technologies*	*Unified Application Development Processes*	*Training of Multiple Categories of Users*
	Monitoring of Management Processes	*Comprehensive Change Management*	*Infrastructure and Connectivity*	*Documented Operational Processes*	*Introduction of Operational Team(s)*
	Key Members within Project Organization	*Disciplined COTS Development Processes*	*Standards Supporting Technology Configuration*		
LESSONS LEARNED	*Develop comprehensive project plans identifying entry and exit milestones for each project phase; including adequate time for addressing test results*	*Implement requirements management processes that enable all future stages of the project*	*The implementation platform for a system should be base-lined and agreed to early in the development life cycle*	*Clearly communicated outcomes based on common processes should be utilized by all teams comprising a system*	*All potential users of a system should be identified early during the project and appropriate training plans developed accordingly*
	Develop project plans using a disciplined approach that lends itself to quantitative measurement for progress	*Use change management processes encompassing the entire system, i.e., software, hardware, integration, training, etc.*	*Software development should always occur in the environment within which the systems will operate*	*Operational processes should be documented and maintained current for development teams to plan ahead*	*Operational team tasked with supporting a new application should be an integral part of the software migration process from development to production*
	Utilize efficient management processes for effective communication and management of expectations	*Institute appropriate software development processes that support effective system development using 4GL (4th generation languages), COTS and COTS (or industry supported) APIs (application programming interfaces)*	*A dedicated technical architect should be a key member of the software development team*		
	Organize and staff project teams identifying clear roles and responsibilities for all key activities				
	Utilization of resources skilled in specific responsibilities assigned to them as well as the experience necessary for required automated tools				